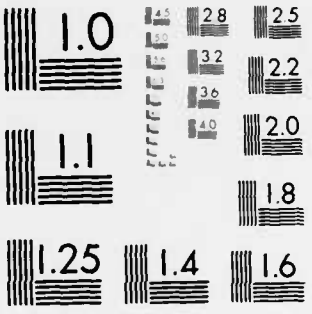
AD-A139 184 A PARALLEL QR METHOD USING FAST GIVENS' ROTATIONS(U) 1/1
YALE UNIV NEW HAVEN CT DEPT OF COMPUTER SCIENCE
I C IPSEN JAN 84 YALEU/DCS/RR-299 N00014-82-K-0184
UNCLASSIFIED F/G 12/1 NL

END
DATE FILMED
4-84
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

AD-A139 184

DTIC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DTIC
ELECTE
MAR 2 2 1984
S D
B

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

84 03 05 012

Abstract

Given a prescribed order in which to introduce zeroes, and constraints on the architecture it is shown how to develop a parallel QR factorisation based on fast Givens' rotations for a rectangular array of processors, suitable to VLSI implementation. Unlike designs based on standard Givens' transformations, the present one requires no square root computations. Assuming each processor performs the elementary operations $(+, *, /)$, less than $O(w^2)$ processors can achieve the decomposition of a w-banded, order n matrix in time $O(n)$.

Application is made to a variant of Bareiss' G-Algorithm for the solution of weighted multiple linear least squares problems. Given k different right hand side vectors, $(w^2 + kw)$ processors compute the factorisation in $O(n + k)$ steps.

A Parallel QR Method Using Fast Givens' Rotations

Ilse C.F. Ipsen

Research Report YALEU/DCS/RR-299
January 1984

DTIC
ELECTE
MAR 2 2 1984
S D
B

The work presented in this paper was supported by the Office of Naval Research under contract N000014-82-K-0184.

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

1. Introduction

Several recent papers, surveyed in [Hl83], demonstrate that a QR decomposition based on Givens' rotations lends itself well to parallel implementation on a rectangular grid of processors in silicon. In [Hl83] it is shown how less than $O(w^2)$ processors accomplish the decomposition of an order n matrix with bandwidth w in time cn, c a small constant. However, all extant designs make use of standard Givens' rotations. Their drawback is the need to compute a square root and $4w$ multiplications for each element to be removed. Multiplication and even more so square root still belong to the most expensive operations, in terms of both chip area and time; their avoidance is desirable.

Fast Givens' rotations are the remedy. Rather than to the original matrix $\bar{A}$, they are applied to a factored form $\bar{A} = DA$, D being a nonsingular diagonal matrix. The square root is obviated and the number of multiplications reduced by fifty percent.

Fast Givens' rotations (their properties are summarised in [Ham74]), as they occur in the context of QR decomposition, are used for

- updating of the solution of linear systems in unconstrained optimisation of homogeneous functions [KK78],
- solution of linear systems in the revised simplex method for linear programming [HW79],
- solution to linear least squares problems [Gen73],
- similarity transformations in the Jacobi method, reduction to Hessenberg form and the QR method for eigenvalue computations [Rat82].

The processor grid is easily extended to handle solutions of linear least squares problems as in [Gen73] with multiple right hand sides. An interesting extension is possible, though. Slightly modifying the processors to compute Bareiss' G-Transformations of order 2, one obtains a processor grid for the solution of weighted least squares problems.

As for the organisation of this paper, a brief description of standard and fast Givens' rotations is followed by some comments on the restrictions placed on parallel architectures by technology. Thereafter the QR algorithm is presented, interconnections and data flow of the processor array are determined and finally applied to the solution of weighted linear least squares problems. The last section remarks on how to extend the design to lower triangular decompositions and matrices the bandwidths of which disagree with the array size.

Householder's notation will be used throughout the paper.

2. Standard and Fast Givens' Rotations

The Givens' plane rotation is a computationally stable device for introducing zeros into a matrix, and it will be illustrated how it inserts a zero in the (2,1) entry a $2 \times n$ matrix, $n \geq 1$.

The standard Givens' rotation [Wilk65], which alters the matrix proper, is a 2×2 transformation

$$P = \begin{pmatrix} \gamma & \sigma \\ -\sigma & \gamma \end{pmatrix}, \qquad \gamma^2 + \sigma^2 = 1,$$

so that

$$\begin{pmatrix} \beta'_{11} & \beta'_{12} & \cdots & \beta'_{1n} \\ 0 & \beta'_{22} & \cdots & \beta'_{2n} \end{pmatrix} = P \begin{pmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1n} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2n} \end{pmatrix}$$

with

$$\delta = \sqrt{\beta_{11}^2 + \beta_{21}^2}, \quad \gamma = \beta_{11}/\delta, \quad \sigma = \beta_{21}/\delta,$$

and

$$\beta'_{1i} = \gamma\beta_{1i} + \sigma\beta_{2i}, \quad \beta'_{2i} = -\sigma\beta_{1i} + \gamma\beta_{2i}.$$

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
PER LETTER
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

To do away with square roots and a major portion of the multiplications, fast Givens' rotations [Gen73, Ham74] modify the matrix in factored form

$$\begin{pmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1n} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2n} \end{pmatrix} = \begin{pmatrix} \delta_1 & 0 \\ 0 & \delta_2 \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \end{pmatrix}.$$

Application of the standard transformation yields

$$\begin{pmatrix} \delta_1' & 0 \\ 0 & \delta_2' \end{pmatrix} \begin{pmatrix} \alpha_{11}' & \alpha_{12}' & \cdots & \alpha_{1n}' \\ 0 & \alpha_{22}' & \cdots & \alpha_{2n}' \end{pmatrix} = P \begin{pmatrix} \delta_1 & 0 \\ 0 & \delta_2 \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \end{pmatrix},$$

or

$$\begin{pmatrix} \delta_1' & 0 \\ 0 & \delta_2' \end{pmatrix} \begin{pmatrix} \alpha_{11}' & \alpha_{12}' & \cdots & \alpha_{1n}' \\ 0 & \alpha_{22}' & \cdots & \alpha_{2n}' \end{pmatrix} = \begin{pmatrix} \gamma\delta_1 & \sigma\delta_2 \\ -\sigma\delta_1 & \gamma\delta_2 \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \end{pmatrix}.$$

Now,

$$\delta = \sqrt{\delta_1^2\alpha_{11}^2 + \delta_2^2\alpha_{21}^2}, \quad \gamma = \delta_1\alpha_{11}/\delta, \quad \sigma = \delta_2\alpha_{21}/\delta,$$

so

$$\delta_1'\alpha_{1i}' = \gamma\delta_1\alpha_{1i} + \sigma\delta_2\alpha_{2i}, \quad \delta_2'\alpha_{2i}' = -\sigma\delta_1\alpha_{1i} + \gamma\delta_2\alpha_{2i}.$$

There are several choices [Hamm74] for δ_1' and δ_2' which permit α_{1i}' and α_{2i}' to be computed with only two multiplications; among them two which - when properly combined - permit easy control of element growth :

$$\begin{pmatrix} \delta_1' & 0 \\ 0 & \delta_2' \end{pmatrix} = \begin{pmatrix} \gamma\delta_1 & 0 \\ 0 & \gamma\delta_2 \end{pmatrix}, \tag{a}$$

and

$$\begin{pmatrix} \delta_1' & 0 \\ 0 & \delta_2' \end{pmatrix} = \begin{pmatrix} \sigma\delta_1 & 0 \\ 0 & \sigma\delta_2 \end{pmatrix}. \tag{b}$$

Regarding case (a),

$$\begin{pmatrix} \alpha_{11}' & \alpha_{12}' & \cdots & \alpha_{1n}' \\ 0 & \alpha_{22}' & \cdots & \alpha_{2n}' \end{pmatrix} = \begin{pmatrix} 1 & \frac{\sigma\delta_2}{\gamma\delta_1} \\ -\frac{\sigma\delta_1}{\gamma\delta_2} & 1 \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \end{pmatrix}.$$

Square roots are avoided by observing that

$$\frac{\sigma\delta_2}{\gamma\delta_1} = \frac{\delta_2^2\alpha_{21}}{\delta_1^2\alpha_{11}}, \quad \frac{\sigma\delta_1}{\gamma\delta_2} = \frac{\alpha_{21}}{\alpha_{11}},$$

and thus one has

$$\begin{pmatrix} \alpha_{11}' & \alpha_{12}' & \cdots & \alpha_{1n}' \\ 0 & \alpha_{22}' & \cdots & \alpha_{2n}' \end{pmatrix} = \begin{pmatrix} 1 & \frac{\delta_2^2\alpha_{21}}{\delta_1^2\alpha_{11}} \\ -\frac{\alpha_{21}}{\alpha_{11}} & 1 \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \end{pmatrix}$$

and

$$\begin{pmatrix} \delta_1' & 0 \\ 0 & \delta_2' \end{pmatrix} = \left(1 + \frac{\delta_2^2\alpha_{21}^2}{\delta_1^2\alpha_{11}^2}\right)^{-1} \begin{pmatrix} \delta_1 & 0 \\ 0 & \delta_2 \end{pmatrix}.$$

Note, that

$$\alpha_{11}' = \left(1 + \frac{\delta_2^2\alpha_{21}^2}{\delta_1^2\alpha_{11}^2}\right)\alpha_{11}.$$

Similarly, one obtains for situation (b)

$$\begin{pmatrix} a'_{11} & a'_{12} & \cdots & a'_{1n} \\ 0 & a'_{22} & \cdots & a'_{2n} \end{pmatrix} = \begin{pmatrix} \frac{\delta_1^2 a_{11}}{\delta_2^2 a_{21}} & 1 \\ -1 & \frac{a_{11}}{a_{21}} \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \end{pmatrix},$$

with

$$a'_{11} = \left(1 + \frac{\delta_1^2 a_{11}^2}{\delta_2^2 a_{21}^2}\right) a_{21},$$

as well as

$$\begin{pmatrix} \delta'_1 & 0 \\ 0 & \delta'_2 \end{pmatrix} = \left(1 + \frac{\delta_1^2 a_{11}^2}{\delta_2^2 a_{21}^2}\right)^{-1} \begin{pmatrix} \delta_2 & 0 \\ 0 & \delta_1 \end{pmatrix}.$$

Pivoting ensures stability :

$$\begin{pmatrix} \delta'_1 & 0 \\ 0 & \delta'_2 \end{pmatrix} \begin{pmatrix} a'_{1i} \\ a'_{2i} \end{pmatrix} = \begin{cases} \begin{pmatrix} \delta_1 & 0 \\ 0 & \delta_2 \end{pmatrix} \begin{pmatrix} a_{1i} \\ a_{2i} \end{pmatrix}, & \text{if } a_{21} = 0 \\ \text{case (a)}, & \text{if } \delta_2^2 a_{21}^2 \leq \delta_1^2 a_{11}^2 \\ \text{case (b)}, & \text{otherwise.} \end{cases}$$

3. Parallel Architecture

The following paragraphs will illustrate the development of a fast-Givens-QR method for a parallel processor device to be implemented in silicon (VLSI for instance). Constraints imposed by technology and fabrication demand regular (and if possible planar) processor interconnections as well as processor communication on a nearest neighbour basis. The obvious choice of structure is a rectangular array of synchronously operating processors. Furthermore, to keep the chip area minimal, processors should perform no other than the elementary operations, addition, multiplication and divsion.

The constraint of *local* data exchange suggests a combination of pipelining and multiprocessing to achieve good processor utilisation and speed up in computation time. Pipelining is efficient for long matrices, which often possess a narrow dense band. The hardware should reflect the features of the matrix : a processor count proportional to the bandwidth, not the order of the matrix. Consequently, I/O occurs by codiagonals rather than rows or columns.

A QR method, chosen with regard to the preceeding thoughts, is presented next; it will be followed by a derivation of data flow and distribution of operations on the processor grid.

4. Parallel QR Decomposition

The QR decomposition of a matrix $\bar{A}$ determines a factorisation

$$\bar{A} = \bar{Q}\bar{R},$$

into an upper triangular matrix $\bar{R}$ and an orthogonal matrix $\bar{Q}$, the product of Givens rotations. For fast Givens' transformations in particular, this takes the form

$$DA = QD'R,$$

given that $\bar{A} = DA$, again R is upper triangular, Q orthogonal and D, D' are nonsingular diagonal matrices.

Matrices to be considered are square, of order n, with a (presumably narrow and dense) band of width

$$w = p + q + 1,$$

$q \geq 0$ being the number of subdiagonals and $p \geq 0$ the number of superdiagonals.

(P1) The QR factorisation preserves the bandwidth w of a matrix A : elimination of $a_{k+i,i}$ causes fill-in $a_{k+i-1,i+w-1}$.

Sameh and Kuck [SK78] proposed a simple elimination strategy which, subject to (P2) below, adheres to regular data flow and local communication : banded matrices are reduced to triangular (banded) form through annihilation of elements by subdiagonals, starting from without towards the main diagonal, while proceeding from top to bottom within a subdiagonal. Formally, if $a_{q+1,1}$ is removed at time $t = 1$ then $a_{q-k+i,i}$ is removed at $t = k + i$, $0 \leq k \leq q-1$, $1 \leq i \leq n-q+k$. In the example below for $q = 2, p = 3, n = 6$, matrix entry $(k+i, i)$ contains the time of removal of (subdiagonal) element $(k+i, i)$.

$$\begin{bmatrix} x & x & x & x & & \\ 2 & x & x & x & x & \\ 1 & 3 & x & x & x & x \\ & 2 & 4 & x & x & x \\ & & 3 & 5 & x & x \\ & & & 4 & 6 & x \end{bmatrix}$$

(P2) Elements are removed by rotations in adjacent planes : elimination of $a_{k+i,i}$ takes place by rotating planes $k+i-1$ and $k+i$.

(P3) Each row, excepting the first and the last, is modified by two successive rotations.

To keep indexing in the algorithms consistent, it is presumed that rows affected by less than two rotations participate in identity transformations. Under the temporary assumption that pivoting is unnecessary, the above strategy leads to the following (still sequential) algorithm, implementing case (a). For each subdiagonal k, $q \geq k \geq 1$, the auxiliary variable Δ_k represents intermediate values of diagonal elements between two rotations (and so do matrix elements with fractional superscripts - remember (P3)); in the hardware implementation it will correspond to a register.

$D^{(q)} = D, \quad A^{(q)} = A;$

for $k = q \ldots 1, \quad \Delta_k = \delta_k^{(q)};$

for $k = q \ldots 1,$

for $i = 1 \ldots n-k,$

{*determine FGR which removes ith element of kth subdiagonal, case (a)*}

$$\pi_2^{(k,i)} = \alpha_{k+i,i}^{(k)} / \alpha_{k+i-1,i}^{(k-\frac{1}{2})}, \quad \pi_1^{(k,i)} = \pi_2^{(k,i)} \left(\delta_{k+i}^{(k)}\right)^2 / \Delta_k^2,$$

$$P^{(k,i)} = \begin{pmatrix} 1 & \pi_1^{(k,i)} \\ -\pi_2^{(k,i)} & 1 \end{pmatrix},$$

$$\tau = 1 + \pi_1^{(k,i)} \pi_2^{(k,i)},$$

{*update ith element of* $(k-1)$*st subdiagonal*}

$$\alpha_{k+i-1,i}^{(k-1)} = \tau \alpha_{k+i-1,i}^{(k-\frac{1}{2})}, \qquad [k+i, i]$$

{*update diagonal matrix*}

$$\begin{pmatrix} \delta_{k+i-1}^{(k-1)} & 0 \\ 0 & \Delta_k \end{pmatrix} = \tau^{-1} \begin{pmatrix} \Delta_k & 0 \\ 0 & \delta_{k+i}^{(k)} \end{pmatrix},$$

for $j = 1 \ldots w-1,$

{*apply FGR to ith element of* $(k-j-1)$*st and* $(k-j)$*th codiagonal*}

$$\begin{pmatrix} \alpha_{k+i-1,i+j}^{(k-1)} \\ \alpha_{k+i,i+j}^{(k-\frac{1}{2})} \end{pmatrix} = P^{(k,i)} \begin{pmatrix} \alpha_{k+i-1,i+j}^{(k-\frac{1}{2})} \\ \alpha_{k+i,i+j}^{(k)} \end{pmatrix}; \qquad [k+i, i+j]$$

$D' = D^{(0)}, \quad R = A^{(0)}.$

Hence,

$$D^{(k-1)} A^{(k-1)} = Q^{(k)} D^{(k)} A^{(k)}, \qquad q \geq k \geq 1,$$

and

$$Q^{(k)} = Q^{(k,n-k)} \ldots Q^{(k,1)},$$

where $Q^{(k,i)}$ stands for the 'proper embedding' of $P^{(k,i)}$ in the $n \times n$ identity matrix.

5. Processor Vector

All iterations of the innermost j-loop, expressed as equations $[k+i, i+j]$, could be executed in parallel, if $P^{(k,i)}$ were to be globally broadcast to at least w processors. As this is to be shunned, the iterations are performed successively in different processors through which $P^{(k,i)}$ is pipelined. Each codiagonal is input to a different processor, starting from the left with the outermost subdiagonal entering processor 1. Thereby due to (P1), a vector of w processors suffices to eliminate the outermost subdiagonal in $2n$ steps (a step is defined as the computation time of the slowest processor).

Consider the example $q = 2$ during the first pass through the k-loop ($k = q = 2$). The computation of D will be disregarded for a moment. Since at least w processors are available let processor j compute equation $[2+i, i+j]$. Thus, for processor 1 to remove $\alpha_{3,1}^{(2)}$ by generating $P^{(2,1)}$ at time $t = \lambda$, it must also contain $\alpha_{2,1}^{(1\frac{1}{2})}$, see Figure 1. In the next time step $\lambda + 1$, $P^{(2,1)}$ is ready to be

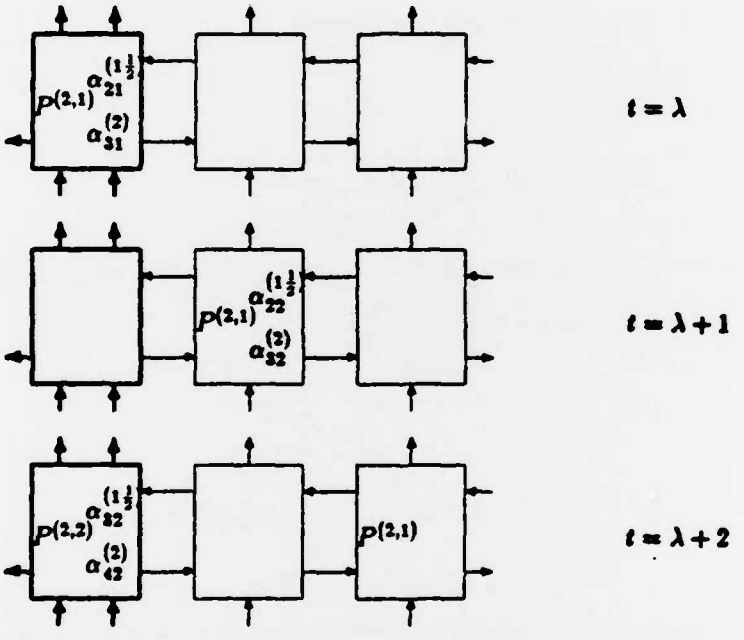


Figure 1: Derivation of Data Flow.

input into processor 2, together with $a_{2,2}^{(1\frac{1}{2})}$ and $a_{3,2}^{(2)}$. Processor 2 then computes equation $[2,2]$, resulting in $a_{2,2}^{(1)}$ and $a_{3,2}^{(1\frac{1}{2})}$. According to (P3), $a_{3,2}^{(1\frac{1}{2})}$ has to participate in a second rotation for the removal of $a_{4,2}^{(2)}$. Consequently, $a_{3,2}^{(2)}$, available at $t = \lambda + 2$, has to enter processor 1 together with $a_{4,2}^{(2)}$ for the determination of $P^{(2,2)}$.

It is now possible to construct the processor vector, depicted in Figure 2. After having entered a processor from below to participate in the first rotation, a matrix element enters the left neighbour for the second rotation, and thereafter exits the top of that processor. Moreover, input to and computation inside a processor occur only every other time step; hence, successive codiagonal elements are separated by one time unit.

In detail, assuming input to the processor vector commences at $t = 1$, then a sub- or maindiagonal element $a_{k+i,i}^{(q)}$ is input to the vector through processor $q-k+1$ at $t = k+2i-1$; it is output from the vector as $a_{k+i,i}^{(q-1)}$ via processor $q-k$ at $t = k+2i$, $0 \le k \le q$. Similarly, a superdiagonal element $a_{i,k+i}^{(q)}$ enters processor $q+k+1$ at $t = k+2i-1$, while $a_{i,k+i}^{(q-1)}$ leaves processor $q+k$ at $t = k+2i$.

The computation of D' remains to be discussed. Since $\delta_{1+i}^{(1)}$ and Δ_2 depend on the same values as $P^{(2,i)}$, they can also be determined in processor 1, but do not have to be broadcast. Δ_2 now denotes a register in processor 1; at the outset it is initialised with δ_2 (δ_1 is not affected by the removal of subdiagonal 2). The contents of Δ_2 are determined during the computation of $P^{(2,i)}$ and kept there till needed for the formation of $P^{(2,i+1)}$. D enters processor 1 along with the 2nd subdiagonal ($\delta_{2+i,i}^{(2)}$ and $a_{2+i,i}^{(2)}$ are input at the same time) and leaves with the, new outermost, 1st subdiagonal ($\delta_{q+i}^{(1)}$ and $a_{q+i,i}^{(1)}$ are output together). Note, that during the annihilation of subdiagonal k, rows $1 \ldots k-1$ of $A^{(k)}$ and $D^{(k)}$ remain unaffected.

6. Processors

Only two different kinds of processors, sketched in Figure 3(a) and (b), will be employed for the triangularisation. The leftmost, first, processor in a vector computes equations $[k+i,i]$, while to its right, processor j determines $[k+i,i+j]$. When idle or no input is available, processor 1 generates identity rotations. The default input for matrix elements is zero.

7. Processor Array

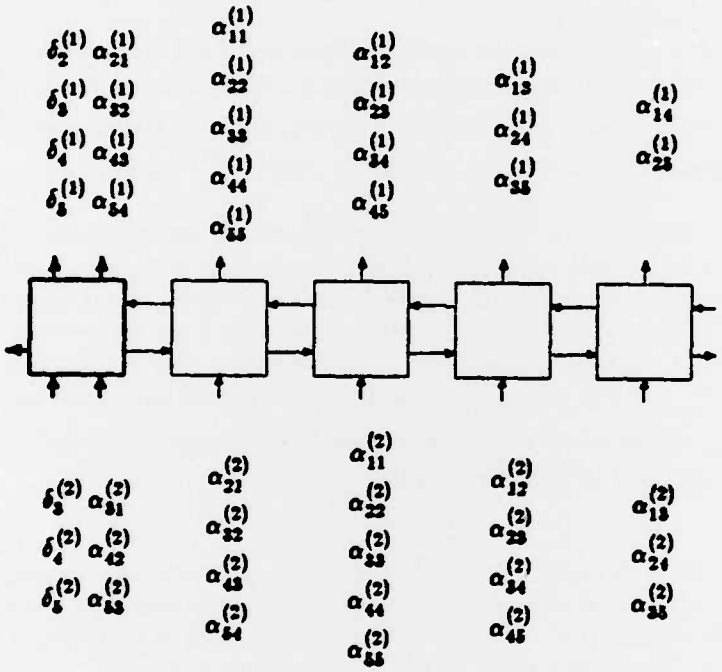
For the annihilation of q subdiagonals, either a single processor vector is used repeatedly (before the $(q-k+1)$st pass register Δ_q in processor 1 is initialised with δ_k, $q \ge k \ge 1$) or else a $q \times w$ array, consisting of q 'stacked' processor vectors, performs the reduction at once.

In the latter case, counting from top to bottom, vector k is responsible for eliminating subdiagonal k. A and D are input to the bottom of the array, while R and D' are available at the top. The matrix Q leaves the right side in factored form. Thus, denoting the jth processor from the left in the kth vector by (k,j), processor $(k,1)$ computes equations $[k+i,i]$ and processor (k,j) equations $[k+i,i+j]$. Figure 4 illustrates the data flow for a $2 \times w$ array when $q = 2$. At the outset of the computation, register Δ_k in processor $(k,1)$ contains δ_k. The computation time comes to $2(n-1+q)$.

8. Weighted Multiple Linear Least Squares Problems

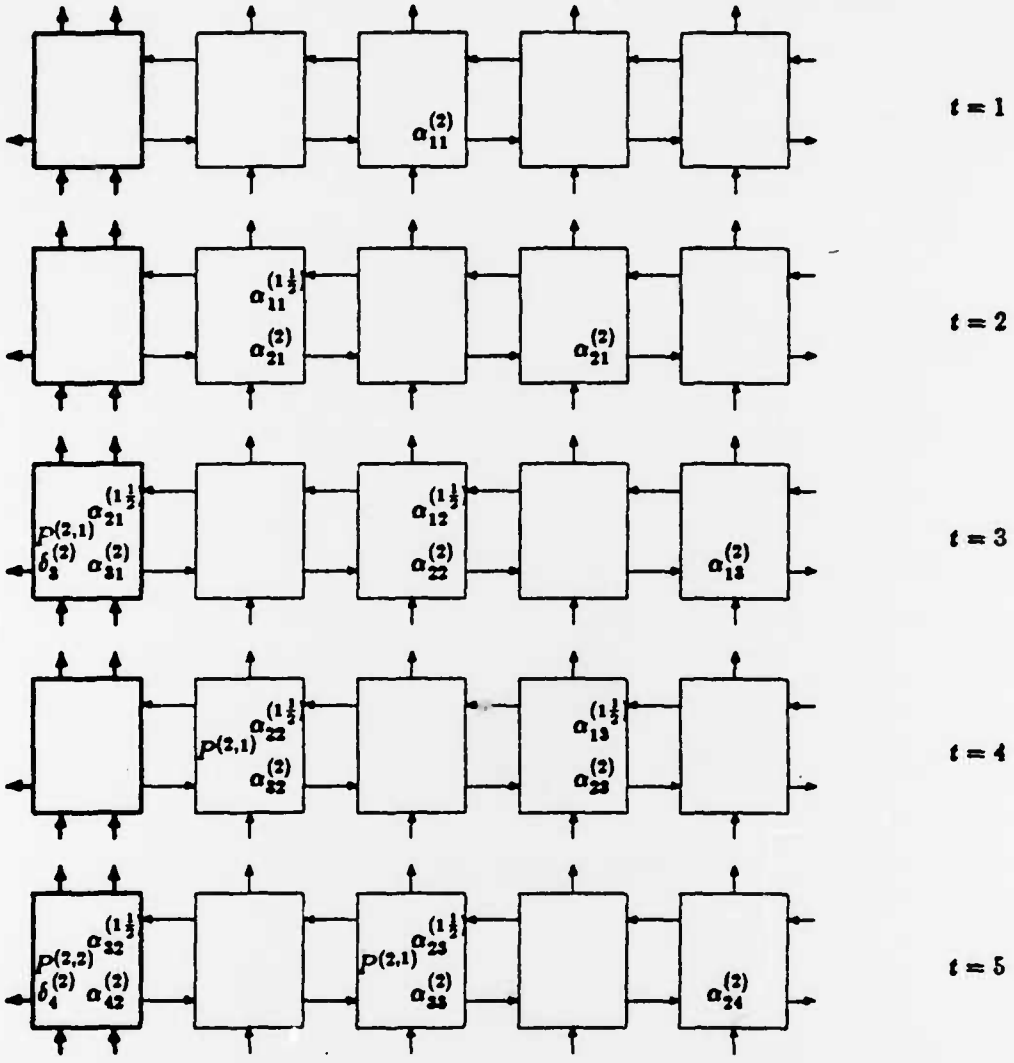
With minor modifications the processor array is adjusted to solve multiple weighted linear least squares problems for $m \times n$ matrices A with $m \ge n$ and $rank(A) = n$,

$$(Ax-b_l)^T W(Ax-b_l) = \|D(Ax-b_l)\|_2 = min, \quad D = W^{\frac{1}{2}}, \quad 1 \le l \le h.$$



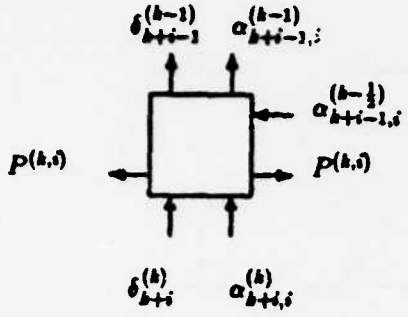
(a) Input and Output.
Processor 1 is initialised with $\delta_2^{(2)}$.

Figure 2: Processor Vector for $w = 5$, Input Matrix with $p = q = 2$.

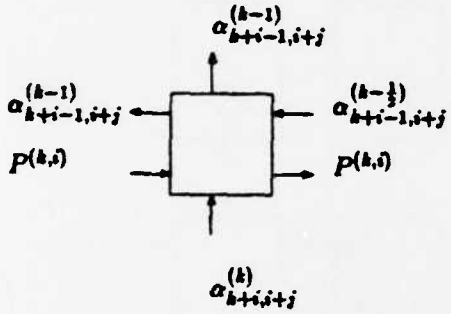


(b) Partial Computation Trace.
Processor 1 is initialised with $\delta_2^{(2)}$.

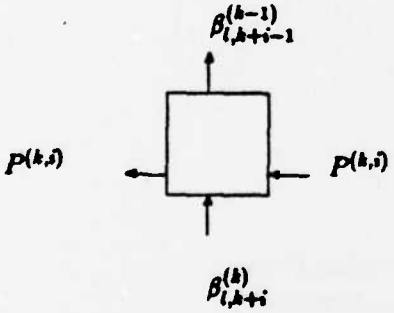
Figure 2: Processor Vector for $w = 5$, Input Matrix with $p = q = 2$.



(a) Equations $[k+i, i]$.



(b) Equations $[k+i, i+j]$.



(c) Equations $[l, k+i]$.

Figure 3: Processors.

The discussion will omit the backsubstitution and focus on the triangular decomposition.

The G-Transformations of Bareiss for weighted linear least squares problems [Bar82] are, like Householder Transformations, intended to remove all elements in a column.

Let

$$B^{(0)} = [A, b_l]$$

and $Q^{(k)}$ an orthogonal transformation that eliminates the last $n-k$ elements in column k of DA, then

$$B^{(k+1)} = Q^{(k+1)} B^{(k)}, \quad 0 \le k \le n-1,$$

and

$$R = B^{(n-1)}, \quad Q = Q^{(n-1)} \dots Q^{(1)}.$$

With nonsingular diagonal matrices $D^{(i)}$,

$$G^{(k)} = \left(D^{(k+1)}\right)^{-1} Q^{(k)} D^{(k)}, \quad 1 \le k \le n-1,$$

so that

$$\left[A^{(k+1)}, b_l^{(k+1)}\right] = D^{(k+1)} G^{(k+1)} \left[A^{(k)}, b_l^{(k)}\right], \quad 0 \le k \le n-1,$$

if

$$\left[A^{(0)}, b_l^{(0)}\right] = [A, b_l].$$

The underling idea is to update $W^{(i)}$ instead of $D^{(i)}$ and thus obviate square roots.

It turns out, however, that matrices $Q^{(k)}$ of order 2 are equal to standard Givens' rotations. *Applying 2×2 matrices $G^{(k)}$ to band matrices in the order prescribed by Sameh and Kuck [SK78],* one obtains a method of the same structure as before. The 2×2 transformations are determined as follows. Let

$$W = \begin{pmatrix} \omega_1 & 0 \\ 0 & \omega_2 \end{pmatrix}, \quad A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \end{pmatrix}, \quad A' = \begin{pmatrix} a'_{11} & a'_{12} & \dots & a'_{1n} \\ 0 & a'_{22} & \dots & a'_{2n} \end{pmatrix},$$

and

$$A' = Q W^{\frac{1}{2}} A,$$

where Q is a standard Givens' rotation

$$Q = \begin{pmatrix} \gamma & \sigma \\ -\sigma & \gamma \end{pmatrix},$$

with

$$\delta = \omega_1 a_{11}^2 + \omega_2 a_{21}^2, \quad \gamma = \sqrt{\frac{\omega_1 a_{11}^2}{\delta}}, \quad \sigma = \sqrt{\frac{\omega_2 a_{21}^2}{\delta}}.$$

Choose

$$W' = \begin{pmatrix} \delta & 0 \\ 0 & \frac{a_{11}\omega_1}{\delta} \end{pmatrix},$$

thus

$$G = (W')^{-\frac{1}{2}} Q W^{\frac{1}{2}} = \begin{pmatrix} \frac{\omega_1 a_{11}}{\delta} & \frac{\omega_2 a_{21}}{\delta} \\ -\frac{a_{21}}{a_{11}} & 1 \end{pmatrix},$$

and

$$A' = (W')^{\frac{1}{2}} G A.$$

Note, that

$$a_{11} = 1.$$

Accordingly, the complete decomposition algorithm is written as

$W^{(q)} = W, \quad A^{(q)} = A;$

for $l = 1 \ldots h, \quad b_l^{(q)} = b_l;$

for $k = q \ldots 1, \quad \Delta_k = \omega_k^{(q)};$

for $k = q \ldots 1,$

  for $i = 1 \ldots n - k,$

    {*determine G to remove ith element of kth subdiagonal*}

$$\delta = \Delta_k \left(a_{k+i-1,i}^{(k-\frac{1}{2})}\right)^2 + \omega_{k+i}^{(k)} \left(a_{k+i,i}^{(k)}\right)^2,$$

$$G^{(k,i)} = \begin{pmatrix} \Delta_k a_{k+i-1,i}^{(k-\frac{1}{2})}/\delta & \omega_{k+i}^{(k)} a_{k+i,i}^{(k)}/\delta, \\ -a_{k+i,i}^{(k)}/a_{k+i-1,i}^{(k-\frac{1}{2})} & 1 \end{pmatrix},$$

    {*update ith element of* $(k-1)$*st subdiagonal*}

$$a_{k+i-1,i}^{(k-1)} = 1, \qquad [k+1, 1]$$

    {*update diagonal matrix*}

$$\begin{pmatrix} \omega_{k+i-1}^{(k-1)} & 0 \\ 0 & \Delta_k \end{pmatrix} = \begin{pmatrix} \delta & 0 \\ 0 & \frac{\omega_{k+i}^{(k)} a_{k+i-1,i}^{(k-\frac{1}{2})}}{\delta} \end{pmatrix},$$

    for $j = 1 \ldots w - 1,$

      {*apply G to ith element of* $(k-j-1)$*st and* $(k-j)$*th codiagonal*}

$$\begin{pmatrix} a_{k+i-1,i+j}^{(k-1)} \\ a_{k+i,i+j}^{(k-\frac{1}{2})} \end{pmatrix} = G^{(k,i)} \begin{pmatrix} a_{k+i-1,i+j}^{(k-\frac{1}{2})} \\ a_{k+i,i+j}^{(k)} \end{pmatrix}; \qquad [k+1, 1+j]$$

    for $l = 1 \ldots h,$

      {*apply G to* $(k+i-1)$*st and* $(k+i)$*th element of lth right hand side vector*}

$$\begin{pmatrix} \beta_{l,k+i-1}^{(k-1)} \\ \Delta_{l,k} \end{pmatrix} = G^{(i,k)} \begin{pmatrix} \Delta_{l,k} \\ \beta_{l,k+i}^{(k)} \end{pmatrix}; \qquad [l, k+1]$$

$W' = W^{(0)}, \quad R = A^{(0)}.$

Again, if the defining elements of the transformation are zero precautions similar to the previous ones have to be taken.

The interconnections and the data flow of the processor array remain, only the processors have to be slightly reprogrammed. In addition, h processor columns of q processors are appended to the left of the array for the computation of equations $[i, k+i]$, resulting in a total of $q(w+k)$ processors. Now, processor $(k,1)$ transmits $G^{(k,i)}$ to its left and right, so $[k+i, i+j]$ and $[l, k+i]$ are computed concurrently. The kth processor (from top) in the lth column (from the right) has a register $\Delta_{l,k}$ which, like Δ_k, enforces (P2); compare Figure 3(c). If the entry equal to one in $G^{(k,i)}$ is implicitly assumed, the transformation can be represented and transmitted by three numbers.

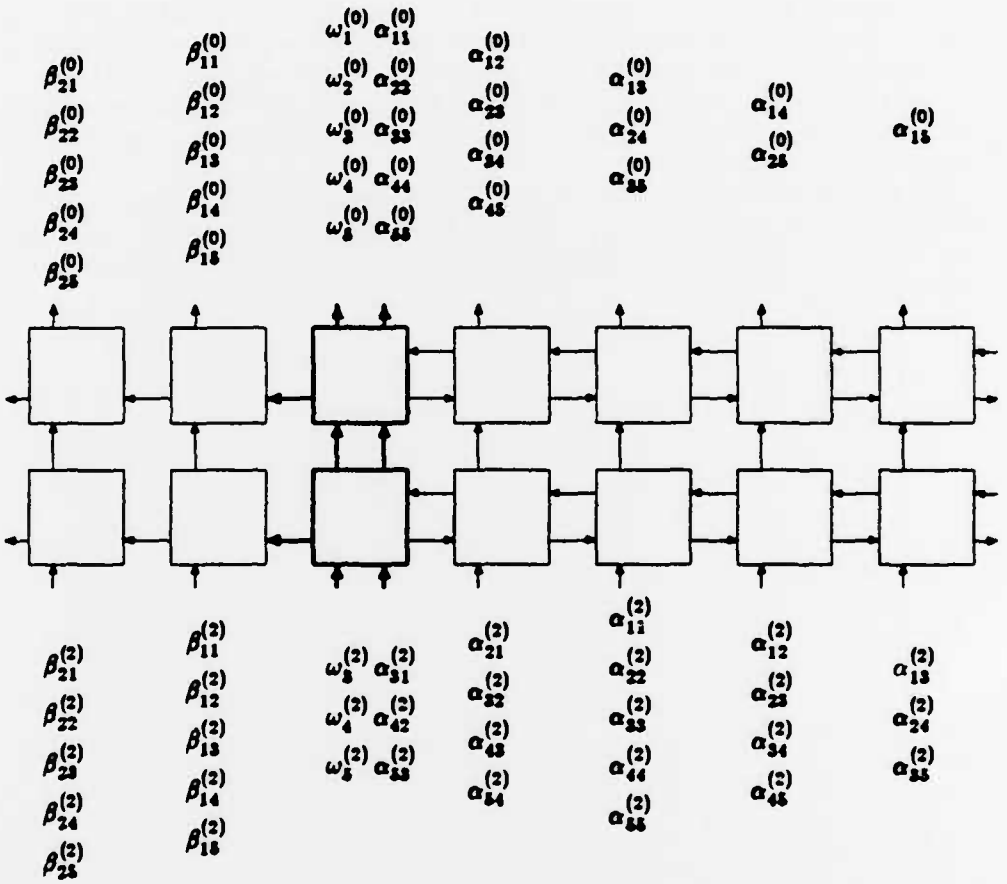
The processor array for $h = p = q = 2$ is depicted in Figure 4. In general, $\beta_{l,1}$ enters $q - l$ steps before a_{11}; and succeeding elements of b_l enter every two steps. $\beta_{11}^{(0)}$ is output j steps after $a_{11}^{(0)}$. Starting the input with β_{11} at $t = 1$, the computation time comes to $2m + 3(q-1) + h$ steps.

9. Remarks

The time to for the QR factorisation of a rectangular $m \times n$ matrix in $q \times w$ array is $2(\max\{m,n\} + q - 1)$. The factorisation of full dense matrices requires $m - 1$ processor vectors of length $m + n - 1$ and time $2(\max\{m,n\} + m - 1)$; hence is less efficient.

A lower tringular, QL decomposition, is obtained by reversing the direction of the horizontal interconnections and computing equations $[k,i]$ in the rightmost processor, as in [HI83] for standard Givens' rotations.

Matrices, whose bandwidth exceeds the length of a processor vector, must be partitioned into smaller submatrices of suitable size. They are then separately , in proper order, input to the array. 'Recycling' of already computed rotations might be necessary. Matrices with too small a bandwidth are input left bound.



Processors $(k,1)$ are initialised with ω_k, $1 \leq k \leq 2$.

Figure 1: Processor Array for the Weighted Multiple Linear Least Squares Problem, $q = p = k = 2$.

1. Bibliography

[Bar82] Bareiss, E.H., "Numerical Solution of the Weighted Linear Least Squares Problem by G-Transformations," *Dept. Comp. Sci. E.E., Northwestern University, Evanston, IL*, 82-03-NAM-03 (1982).

[Gen73] Gentleman, W.M., "Least Squares Computations by Givens Transformations without Square roots," *J. Inst. Math. Appl. 12*, 329-36 (1973).

[Ham74] Hammarling, S., "A Note on Modifications to the Givens Plane Rotation," *J. Inst. Math. Appl. 13*, 215-18 (1974).

[HW79] Hanson, R.J. and Wisniewski, J.A., "A Mathematical Programming Updating Method Using Modified Givens Transformations and Applied to LP Problems," *CACM 22*, 245-51 (1979).

[Hl83] Heller, D.E. and Ipsen, I.C.F, "Systolic Networks for Orthogonal Decompositions," *Siam J. Sci. Stat. Comp. 4*, 261-9 (1983).

[KK78] Kowalik, J.S. and Kumar, S.P., "Fast Givens Transformations Applied to the Homogeneous Optimization Method," *Appl. Math. Comp. 4*, 239-52 (1978).

[Rat82] Rath, W., "Fast Givens Rotations for Orthogonal Similarity Transformations," *Num. Math. 40*, 47-56 (1982).

[SK78] Sameh, A.H. and Kuck, D.J, "On Stable Linear System Solvers," *JACM 25*, 81-91 (1978).


DATE
ILME

thus

$$G = (W')^{-\frac{1}{2}} Q W^{\frac{1}{2}} = \begin{pmatrix} \frac{\omega_1 \sigma_{11}}{\delta} & \frac{\omega_2 \sigma_{21}}{\delta} \\ -\frac{\sigma_{21}}{\sigma_{11}} & 1 \end{pmatrix},$$

and

$$A' = (W')^{\frac{1}{2}} GA.$$

to the left of the array for the computation of equations $[i, k+i]$, resulting in a total of $q(w+k)$ processors. Now, processor $(k,1)$ transmits $G^{(k,i)}$ to its left and right, so $[k+i, i+j]$ and $[l, k+i]$ are computed concurrently. The kth processor (from top) in the lth column (from the right) has a register $\Delta_{l,k}$ which, like Δ_k, enforces (P2); compare Figure 3(c). If the entry equal to one in $G^{(k,i)}$ is implicitly assumed, the transformation can be represented and transmitted by three numbers.